NYT Prediction Report

Exploring Data

There are 8402 observations in total, where 6532 training observations and 1870 observations.

```
popularDensity <- table(newsData$Popular)
posPopular <- round(popularDensity[2]/(popularDensity[1] + popularDensity[2]) * 100, 2)
```

Only 16.73% of all New York Times blog articles have more than 25 comments. That means, a baseline model for predicting unpopular would be around 83.27%.

The independent variables consist of 8 pieces of article data available at the time of publication, and a unique identifier:

- **NewsDesk**, the New York Times desk that produced the story (Business, Culture, Foreign, etc.)
- **SectionName**, the section the article appeared in (Opinion, Arts, Technology, etc.)
- **SubsectionName**, the subsection the article appeared in (Education, Small Business, Room for Debate, etc.)
- **Headline**, the title of the article
- **Snippet**, a small portion of the article text
- **Abstract**, a summary of the blog article, written by the New York Times
- **WordCount**, the number of words in the article
- **PubDate**, the publication date, in the format "Year-Month-Day Hour:Minute:Second"
- **UniqueID**, a unique identifier for each article

Cleaning Data

Text of Articles

Let's take a look at **Headline**, we can observe many very common combination of words like `new york times`, `pictures of the day` etc. If we google `pictures of the day new york times`, it is easy to know `pictures of the day` is a daily article from `Lens` category.

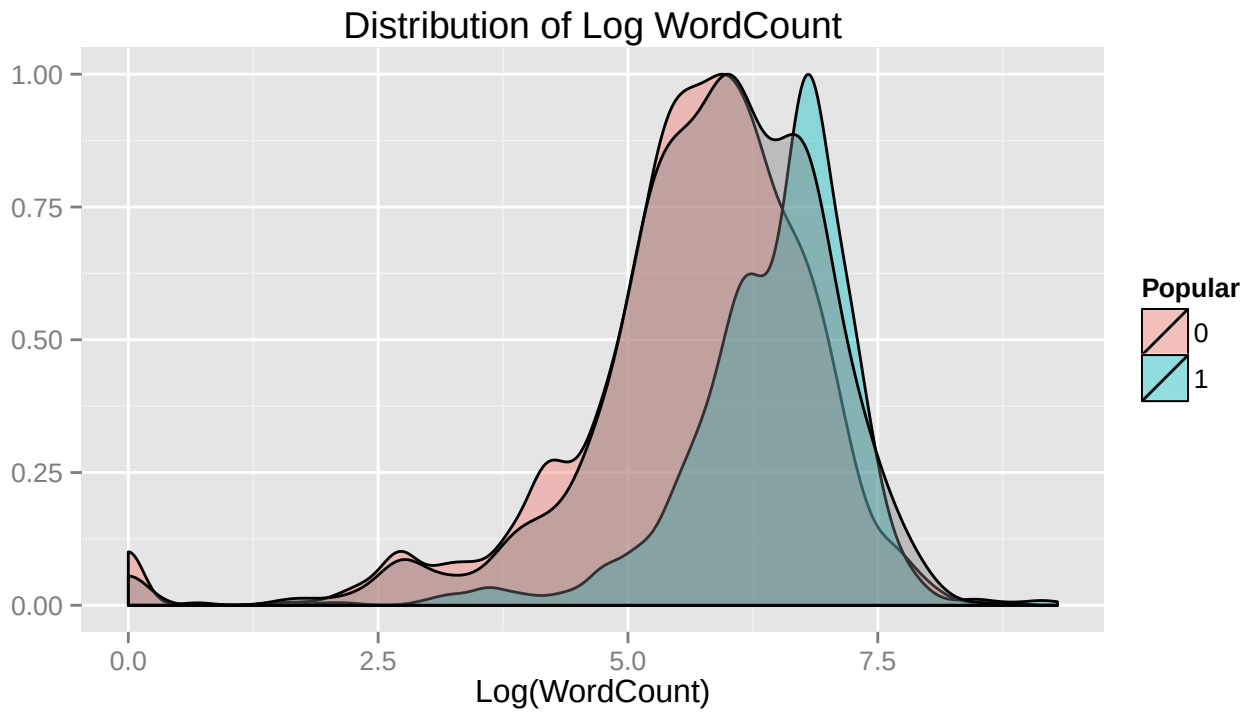
```
newsData$Headline
```

To avoid overcounting of words like `day`, a replacement of some proper nouns to single word is necessary.

```
originalText    = c("new york times", "new york city", "new york", "silicon valley",
                    "times insider", "fashion week", "white house",
                    "international herald tribune archive",
                    "president obama", "hong kong", "big data", "golden globe",
                    "word of the day", "time square", "pictures of the day",
                    "photos of the day", "daily clip report", "daily report",
                    "6 q's about the news", "test yourself")

replacementText = c("NYT", "NYC", "NewYork", "SiliconValley", "TimesInsider",
                    "FashionWeek", "WhiteHouse", "IHT", "Obama", "HongKong",
                    "BigData", "GoldenGlobe", "WordofDay", "TimeSquare", "PicOfDay",
                    "PicOfDay", "DailyClipReport", "DailyReport", "6q", "TestYourself")
```

Word Count of Articles

The following plot shows article's popularity distribution based on logarithmic word count. Beside training data, testing data's distribution is also plotted by gray color which is bimodal distribution.



If we conduct a two-sided t-test on the mean and a two-sided F-test on the variance:

```
PopularNewsTrain   = subset(newsTrain, newsTrain$Popular==1)
UnpopularNewsTrain = subset(newsTrain, newsTrain$Popular==0)

t.test(PopularNewsTrain$LogWordCount, UnpopularNewsTrain$LogWordCount)
```

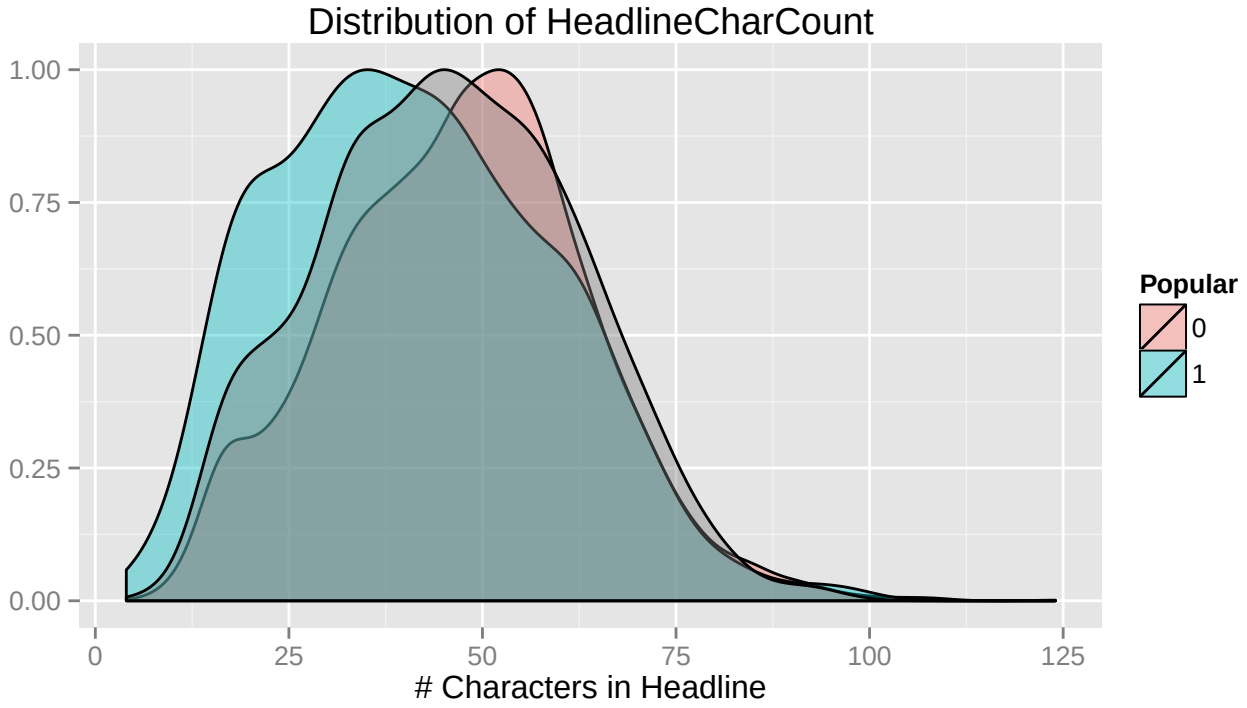
```
##
##  Welch Two Sample t-test
##
## data:  PopularNewsTrain$LogWordCount and UnpopularNewsTrain$LogWordCount
## t = 28.2691, df = 2310.719, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.8018917 0.9214367
## sample estimates:
## mean of x mean of y
##  6.455548  5.593884
```

```
var.test(PopularNewsTrain$LogWordCount, UnpopularNewsTrain$LogWordCount)
```

```
##
##  F test to compare two variances
##
```

```
## data:  PopularNewsTrain$LogWordCount and UnpopularNewsTrain$LogWordCount
## F = 0.4108, num df = 1092, denom df = 5438, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.3752262 0.4509859
## sample estimates:
## ratio of variances
##          0.4107796
```

This shows us there is a statistically significant difference between popular and unpopular articles based on the word counts. At the same time, popular article seems having shorter `Headline`.




```
t.test(PopularNewsTrain$HeadlineCharCount, UnpopularNewsTrain$HeadlineCharCount)
```

```
##
##  Welch Two Sample t-test
##
## data:  PopularNewsTrain$HeadlineCharCount and UnpopularNewsTrain$HeadlineCharCount
## t = -10.4608, df = 1483.142, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -7.191321 -4.920218
## sample estimates:
## mean of x mean of y
##  41.16651  47.22228
```

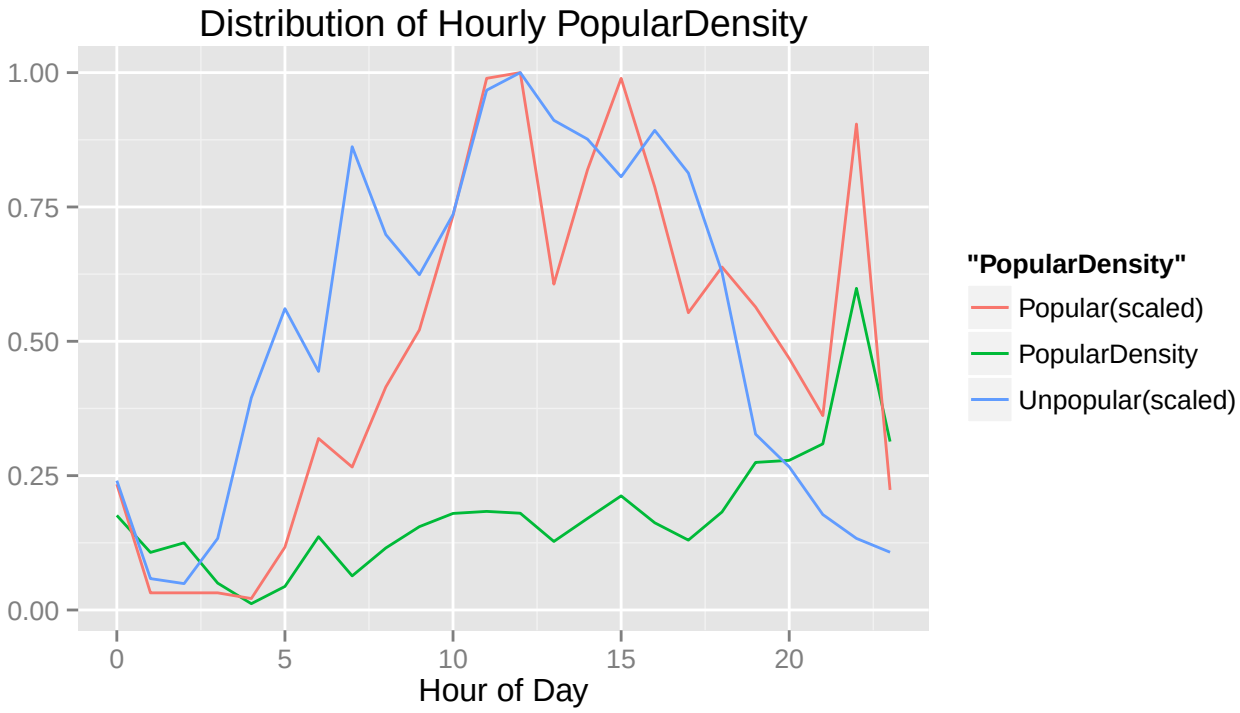
Publishing Hour and Day

It is unlikely that many article receiving 25 more comments in the middle of night. Hence, at certain times during the day, we expect the probability that a random article becomes popular to be larger. Similarly, the day of the week may have an impact, people may have much more time to read articles than a working day.

```
## date feature
newsData$PubDate = strptime(newsData$PubDate, "%Y-%m-%d %H:%M:%S")
newsData$PubDay  = as.Date(newsData$PubDate)
## it is expected that different behaviours at different times of the day.publication
newsData$DayofWeek = newsData$PubDate$wday
newsData$Hour    = newsData$PubDate$hour
newsData$DayOfWeek = as.factor(weekdays(newsData$PubDate))
newsData$DayOfWeek = factor(newsData$DayOfWeek, levels=c("Monday", "Tuesday", "Wednesday", "Thursday",

## especially on holidays, people may have much more time to read and comment on blog articles
Holidays = c(as.POSIXlt("2014-09-01 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-10-13 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-10-31 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-11-11 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-11-27 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-12-24 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-12-25 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-12-31 00:00", format="%Y-%m-%d %H:%M"))

newsData$Holiday = as.factor(ifelse(newsData$PubDate$yday %in% Holidays$yday, 1, 0))
```

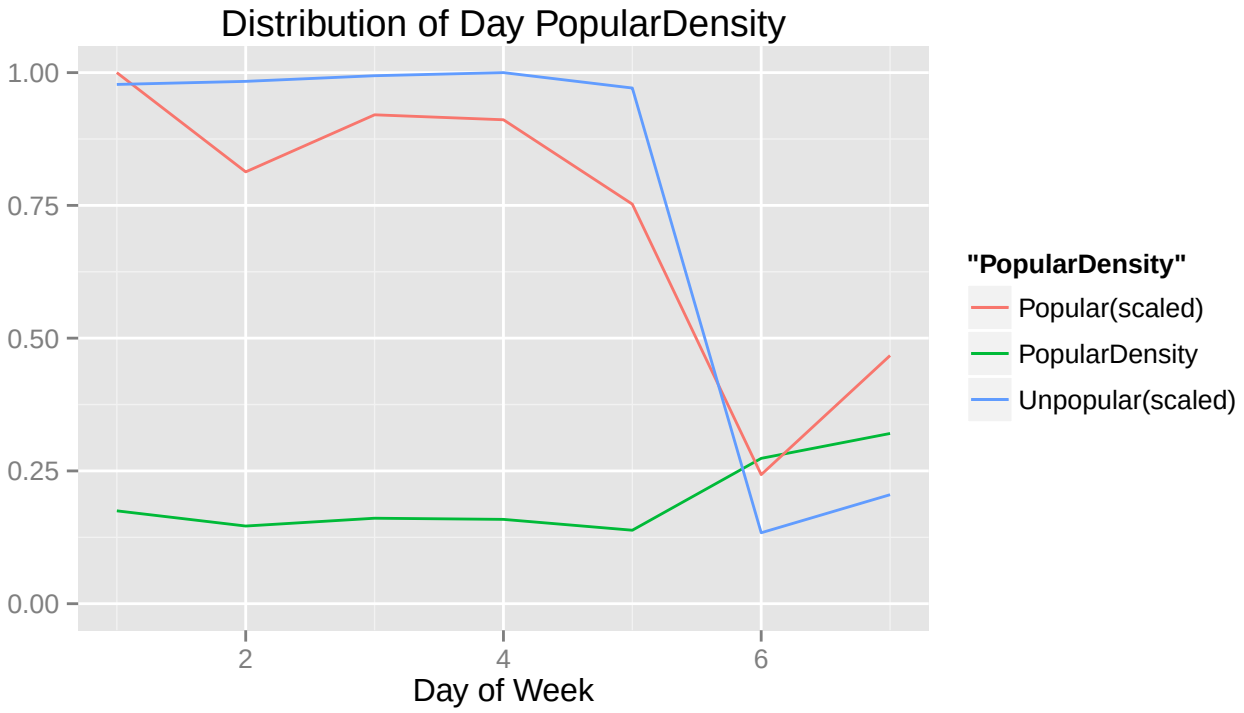
```
##    Unpopular Popular PopularDensity Hour
## 4        169       2     0.01169591    4
## 5        240      11     0.04382470    5
## 3         57       3     0.05000000    3
## 7        369      25     0.06345178    7
## 1         25       3     0.10714286    1
## 8        299      39     0.11538462    8
## 2         21       3     0.12500000    2
## 13       390      57     0.12751678   13
## 17       348      52     0.13000000   17
## 6        190      30     0.13636364    6
## 9        267      49     0.15506329    9
## 16       382      74     0.16228070   16
## 14       375      77     0.17035398   14
## 0        103      22     0.17600000    0
## 10       315      69     0.17968750   10
## 12       428      94     0.18007663   12
## 18       269      60     0.18237082   18
## 11       414      93     0.18343195   11
## 15       345      93     0.21232877   15
## 19       140      53     0.27461140   19
## 20       114      44     0.27848101   20
## 21        76      34     0.30909091   21
## 23        46      21     0.31343284   23
## 22        57      85     0.59859155   22
```

It seems that publishing blog posts around `10 pm` are more easier getting popular according to `PopularDensity`. But, if we compare number of popular articles of 24 hours, It is clear to see that around `12 pm` and `3 pm`, even more blog posts receiving 25 more comments than `10 pm`.




```
##           Unpopular Popular PopularDensity Day
## Friday         1003     161      0.1383162   5
## Tuesday        1016     174      0.1462185   2
## Thursday       1033     195      0.1587948   4
## Wednesday      1027     197      0.1609477   3
## Monday         1010     214      0.1748366   1
## Saturday        138      52      0.2736842   6
## Sunday          212     100      0.3205128   7
```

Similarly, day of week shows same trends as hourly results. Also, much more articles are published on weekday than weekends.



Category of Articles

There are three variables categorizes blog posts, `NewsDesk` `SectionName` and `SubsectionName`.

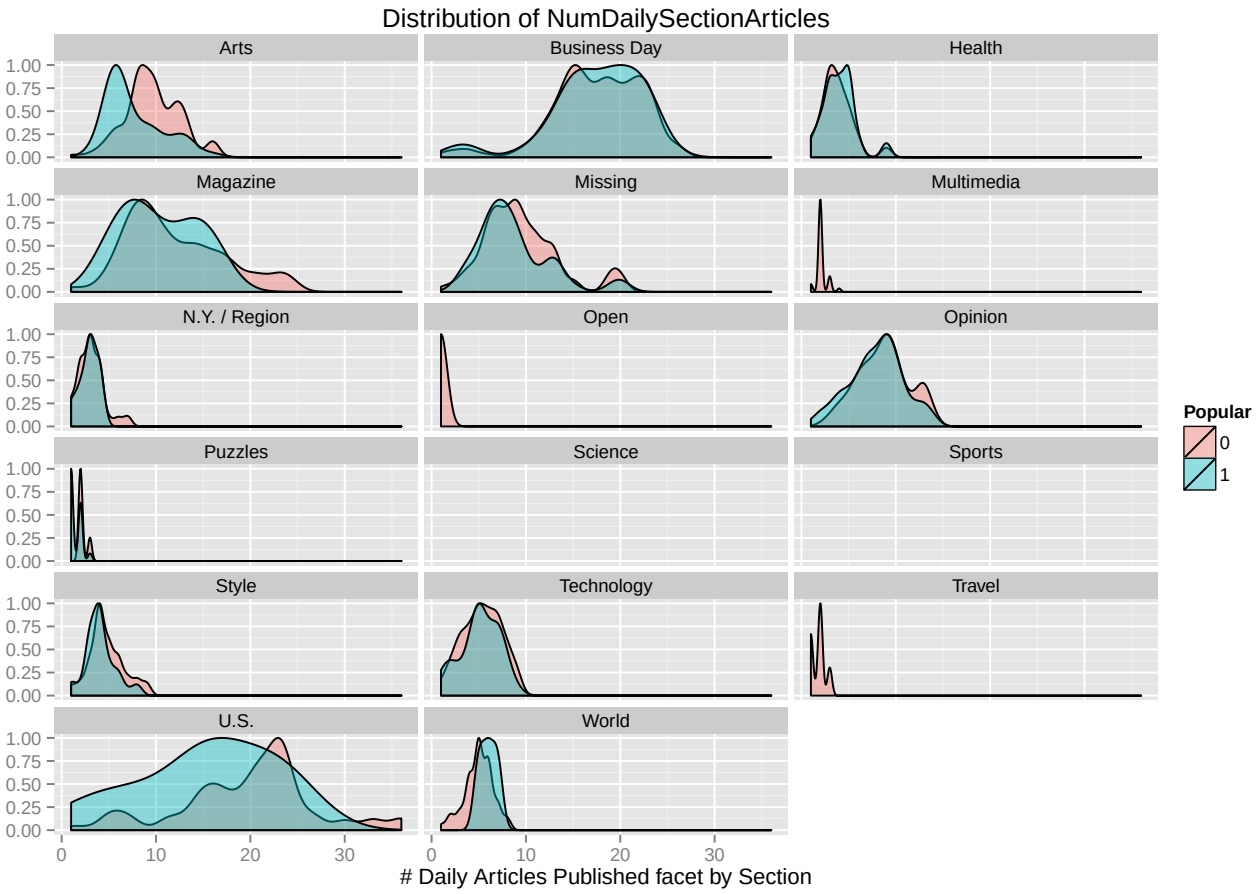
```
## missing categories
misCategory = subset(newsData, newsData$NewsDesk=="" | newsData$SectionName=="" | newsData$SubsectionNa
dim(misCategory)[1]
```

```
## [1] 6721
```

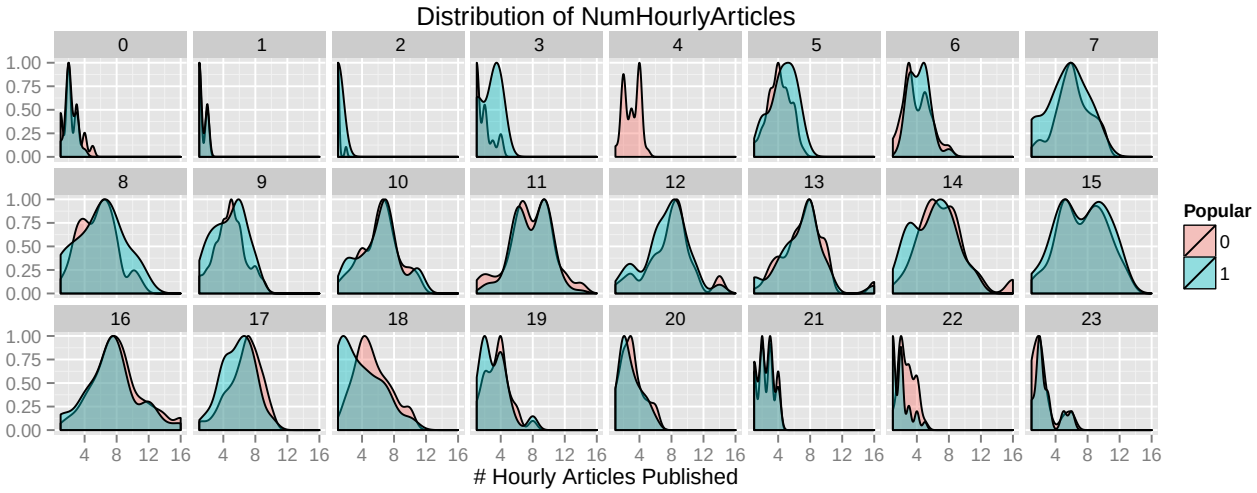
```
misCategory = subset(newsData, newsData$NewsDesk=="" & newsData$SectionName=="" & newsData$SubsectionNa
dim(misCategory)[1]
```

```
## [1] 1626
```

6721 articles have at one category variable missing and **1626** articles have no categories at all. After filling blank categories based on existing category variables, let's try to see the facet distribution of blog posts.



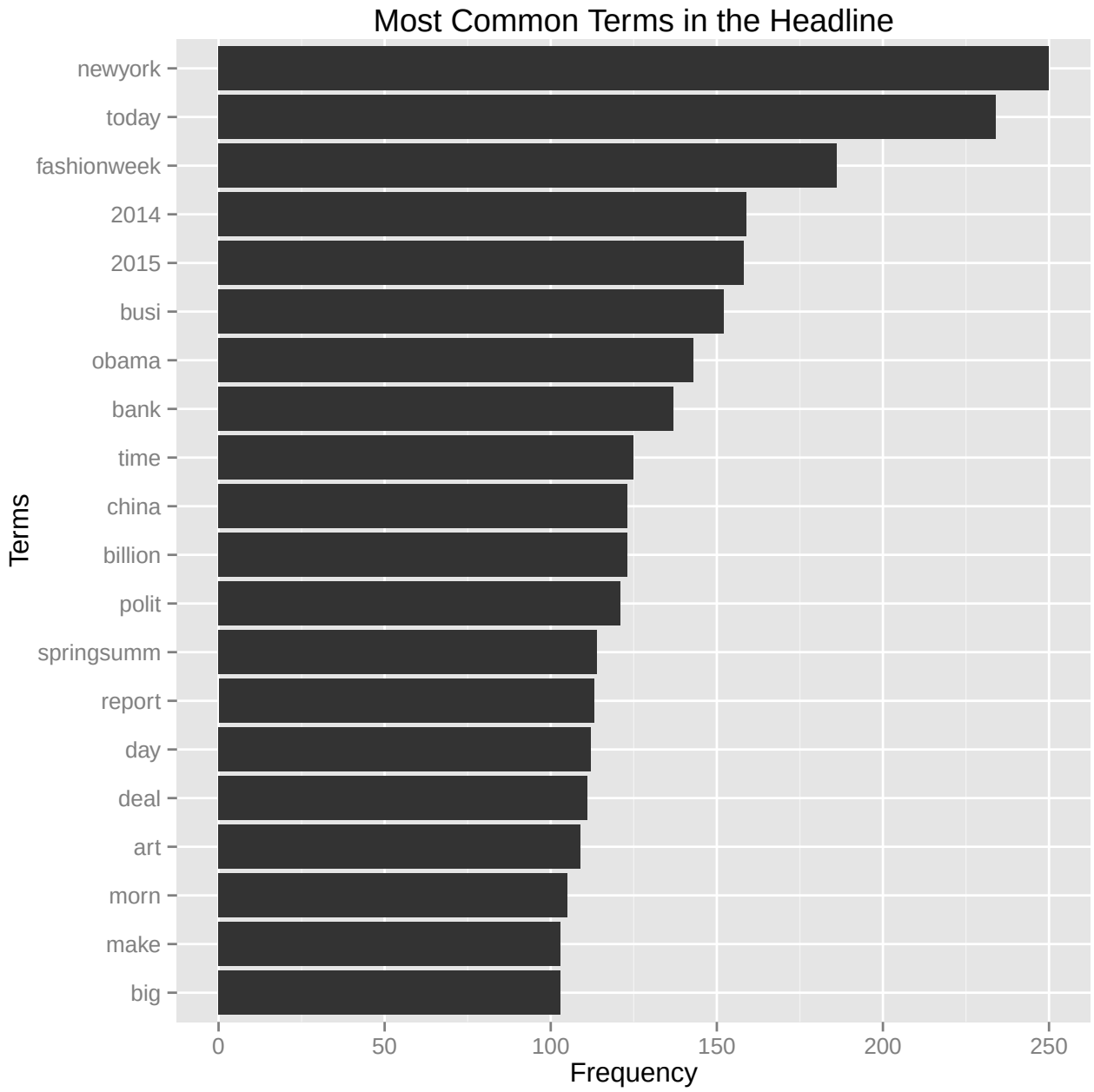
Although it is hard to see what's going on, a clear difference between popular and unpopular articles is in the section Magazine, where around 15 articles posted per day is more indicative of popular articles than unpopular ones. Beyond 20 posts per day the roles are clearly reversed. Hourly distribution of the following plot also shows no clear indication of popularity.



Contents Features of Articles

Until now, we preprocessed date features, word counts and categories of article. Almost all features from original data frame but the contents of blog post.

```
stopWords = c(stopwords("SMART"))
CorpusText = Corpus(VectorSource(newsData$Headline))
CorpusText = tm_map(CorpusText, tolower)
CorpusText = tm_map(CorpusText, PlainTextDocument)
CorpusText = tm_map(CorpusText, removePunctuation)
CorpusText = tm_map(CorpusText, removeWords, stopWords)
CorpusText = tm_map(CorpusText, stemDocument, language="english")

tdmText = TermDocumentMatrix(CorpusText)
# around 20 words
sparseText = removeSparseTerms(tdmText, 0.989)
sparseText = as.data.frame(as.matrix(sparseText))
colnames(sparseText) = make.names(colnames(sparseText))

dtmText = DocumentTermMatrix(CorpusText)
freqTerms = findFreqTerms(dtmText, lowfreq=10)
termFreq  = colSums(as.matrix(dtmText))
termFreq  = subset(termFreq, termFreq>=100)
df        = data.frame(term=names(termFreq), freq=termFreq)

## Bag of Words
newsDataBoW = newsData
newsDataBoW$PubDate = NULL

tSparseText = t(sparseText)
colnames(tSparseText) = make.names(paste('c',colnames(tSparseText),sep="_"))
newsDataBoW[, colnames(tSparseText)] = tSparseText
```



Modeling Data

Logistic Regression without contents feature

```
# MODEL--0
# LR_0
# finding importance of features
removedColumns = c("SectionName", "NewsDesk", "SubsectionName", "Headline", "Snippet", "Abstract", "Sum
LR_0 = glm(Popular ~ ., data=newsTrain[,!colnames(newsTrain) %in% removedColumns], family=binomial)

calcAUClr(LR_0, newsTrain$Popular)
```

```
## [1] 0.9015615 0.9315814
```

```r
# training result: 0.9315814
LR_0_Pred = predict(LR_0, newdata=newsTest[, !colnames(newsTrain) %in% removedColumns], type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
generateSubmission(LR_0_Pred, submitPath)
# testing  result: 0.89088

summary(LR_0)
```

```
##
## Call:
## glm(formula = Popular ~ ., family = binomial, data = newsTrain[,
##     !colnames(newsTrain) %in% removedColumns])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8005  -0.3645  -0.1895  -0.0059   3.4498
##
## Coefficients: (12 not defined because of singularities)
##                                    Estimate Std. Error z value
## (Intercept)                       3.885e+00  4.792e+02   0.008
## PubDay                            1.131e-04  1.940e-03   0.058
## Hour                              3.572e-02  9.917e-03   3.601
## WordCount                         3.052e-04  1.462e-04   2.088
## HeadlineCharCount                -1.409e-02  8.095e-03  -1.741
## SummaryCharCount                 -1.544e-02  4.066e-03  -3.797
## HeadlineWordCount                 1.080e-01  4.170e-02   2.589
## SummaryWordCount                  7.890e-02  2.350e-02   3.357
## LogWordCount                      8.855e-01  1.073e-01   8.253
## ShortHeadline1                    8.897e-02  1.647e-01   0.540
## DayofWeek                        -5.135e-01  3.607e-01  -1.424
## DayOfWeekTuesday                  1.740e-01  4.397e-01   0.396
## DayOfWeekWednesday                8.961e-01  7.822e-01   1.146
## DayOfWeekThursday                 1.272e+00  1.128e+00   1.128
## DayOfWeekFriday                   1.625e+00  1.500e+00   1.083
## DayOfWeekSaturday                 2.324e+00  2.123e+00   1.095
## DayOfWeekSunday                          NA         NA      NA
## Holiday1                         -9.462e-02  2.385e-01  -0.397
## SectionNameFactorBusiness Day    -1.397e+01  4.781e+02  -0.029
## SectionNameFactorHealth          -1.547e+01  6.523e+03  -0.002
## SectionNameFactorMagazine        -1.593e+00  3.993e-01  -3.990
## SectionNameFactorMissing          2.868e-01  1.159e+00   0.247
## SectionNameFactorMultimedia      -2.263e+00  1.379e+00  -1.641
## SectionNameFactorN.Y. / Region    6.207e-02  3.405e-01   0.182
## SectionNameFactorOpen            -1.582e+01  3.104e+03  -0.005
## SectionNameFactorOpinion          4.031e+00  2.187e-01  18.430
## SectionNameFactorPuzzles          3.368e+00  3.537e-01   9.523
## SectionNameFactorScience          2.554e+00  7.987e+03   0.000
## SectionNameFactorSports          -1.827e+01  4.607e+03  -0.004
## SectionNameFactorStyle            1.583e+00  2.384e-01   6.641
```

```
## SectionNameFactorTechnology            7.135e-01  2.506e-01   2.847
## SectionNameFactorTravel               -1.809e+00  1.030e+00  -1.757
## SectionNameFactorU.S.                 -1.520e+01  3.599e+03  -0.004
## SectionNameFactorWorld                -1.536e+01  4.781e+02  -0.032
## NewsDeskFactorCulture                        NA         NA      NA
## NewsDeskFactorForeign                        NA         NA      NA
## NewsDeskFactorMagazine                -1.407e+01  1.088e+03  -0.013
## NewsDeskFactorMetro                          NA         NA      NA
## NewsDeskFactorMissing                 5.706e-01  1.156e+00   0.494
## NewsDeskFactorNational                       NA         NA      NA
## NewsDeskFactorOpEd                           NA         NA      NA
## NewsDeskFactorScience                 1.831e+01  6.523e+03   0.003
## NewsDeskFactorSports                         NA         NA      NA
## NewsDeskFactorStyles                         NA         NA      NA
## NewsDeskFactorTechnology                     NA         NA      NA
## NewsDeskFactorTravel                         NA         NA      NA
## NewsDeskFactorTStyle                         NA         NA      NA
## SubsectionNameFactorDealbook          1.011e+00  4.785e-01   2.114
## SubsectionNameFactorEducation        -1.351e+01  3.646e+03  -0.004
## SubsectionNameFactorFashion & Style  -3.108e+01  4.636e+03  -0.007
## SubsectionNameFactorMissing          -1.330e+01  4.781e+02  -0.028
## SubsectionNameFactorPolitics          1.799e+00  3.631e+03   0.000
## SubsectionNameFactorRoom For Debate  -2.134e+01  4.781e+02  -0.045
## SubsectionNameFactorSmall Business           NA         NA      NA
## SubsectionNameFactorThe Public Editor -1.369e+01  4.781e+02  -0.029
## NumDailyArticles                      4.342e-03  4.175e-03   1.040
## NumDailySectionArticles              -4.238e-02  1.550e-02  -2.735
## NumHourlyArticles                    -5.318e-02  1.820e-02  -2.922
##                                      Pr(>|z|)
## (Intercept)                          0.993531
## PubDay                               0.953512
## Hour                                 0.000316 ***
## WordCount                            0.036821 *
## HeadlineCharCount                    0.081762 .
## SummaryCharCount                     0.000147 ***
## HeadlineWordCount                    0.009638 **
## SummaryWordCount                     0.000787 ***
## LogWordCount                          < 2e-16 ***
## ShortHeadline1                       0.589008
## DayofWeek                            0.154528
## DayOfWeekTuesday                     0.692276
## DayOfWeekWednesday                   0.251948
## DayOfWeekThursday                    0.259461
## DayOfWeekFriday                      0.278668
## DayOfWeekSaturday                    0.273643
## DayOfWeekSunday                            NA
## Holiday1                             0.691587
## SectionNameFactorBusiness Day        0.976684
## SectionNameFactorHealth              0.998108
## SectionNameFactorMagazine            6.60e-05 ***
## SectionNameFactorMissing             0.804539
## SectionNameFactorMultimedia          0.100866
## SectionNameFactorN.Y. / Region       0.855372
## SectionNameFactorOpen                0.995935
```

```
## SectionNameFactorOpinion               < 2e-16 ***
## SectionNameFactorPuzzles               < 2e-16 ***
## SectionNameFactorScience              0.999745
## SectionNameFactorSports               0.996837
## SectionNameFactorStyle                3.12e-11 ***
## SectionNameFactorTechnology           0.004415 **
## SectionNameFactorTravel               0.078882 .
## SectionNameFactorU.S.                 0.996631
## SectionNameFactorWorld                0.974370
## NewsDeskFactorCulture                       NA
## NewsDeskFactorForeign                       NA
## NewsDeskFactorMagazine                0.989684
## NewsDeskFactorMetro                         NA
## NewsDeskFactorMissing                 0.621492
## NewsDeskFactorNational                      NA
## NewsDeskFactorOpEd                          NA
## NewsDeskFactorScience                 0.997760
## NewsDeskFactorSports                        NA
## NewsDeskFactorStyles                        NA
## NewsDeskFactorTechnology                    NA
## NewsDeskFactorTravel                        NA
## NewsDeskFactorTStyle                        NA
## SubsectionNameFactorDealbook          0.034544 *
## SubsectionNameFactorEducation         0.997043
## SubsectionNameFactorFashion & Style   0.994651
## SubsectionNameFactorMissing           0.977800
## SubsectionNameFactorPolitics          0.999605
## SubsectionNameFactorRoom For Debate   0.964404
## SubsectionNameFactorSmall Business          NA
## SubsectionNameFactorThe Public Editor 0.977166
## NumDailyArticles                      0.298318
## NumDailySectionArticles               0.006239 **
## NumHourlyArticles                     0.003480 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5900.1  on 6531  degrees of freedom
## Residual deviance: 3093.0  on 6486  degrees of freedom
## AIC: 3185
##
## Number of Fisher Scoring iterations: 17
```

By using `summary(Model)`, we can select the following variables based on importance of variables.

```
selectedColumns = c("Popular",
                    "PubDay",
                    "Hour",
                    "LogWordCount",
                    "SummaryCharCount",
                    "HeadlineWordCount",
                    "WordCount",
                    "SummaryCharCount",
```

```
                    "HeadlineCharCount",
                    "SectionNameFactor",
                    "SubsectionNameFactor",
                    "NumDailySectionArticles",
                    "NumHourlyArticles")
```

MODEL-1 (logistic regression)

```
LR_1 = glm(Popular ~ ., data=newsTrain[, colnames(newsTrain) %in% selectedColumns], family=binomial)

calcAUClr(LR_1, newsTrain$Popular)
# training result: 0.9297705

LR_1_Pred = predict(LR_1, newdata=newsTest[, colnames(newsTrain) %in% selectedColumns], type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
generateSubmission(LR_1_Pred, submitPath)
# testing  result: 0.89212
```

MODEL-2 (random forest)

```
RF_2 = randomForest(Popular ~ ., data=newsTrain[, colnames(newsTrain) %in% selectedColumns], nodesize=1

trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, colnames(newsTrain) %in% selectedColumns]
RF_2.tuned  = train(Popular ~ ., data=tuneTrain, method="rf", trControl=trainControl(method="cv", numbe

calcAUC(RF_2, newsTrain$Popular)
# training result: 0.9387521

RF_2_Pred = predict(RF_2, newdata=newsTest, type="prob")[,2]
generateSubmission(RF_2_Pred, submitPath)
# testing  result: 0.89470
```

MODEL-3 (logistic regression on Bag of Words)

```
LR_3 = glm(Popular ~ ., data=newsTrainBoW[,!colnames(newsTrainBoW) %in% removedColumns], family=binomial
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
calcAUClr(LR_3, newsTrainBoW$Popular)
# training result: 0.9389116

LR_3_Pred = predict(LR_3, newdata=newsTestBoW, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

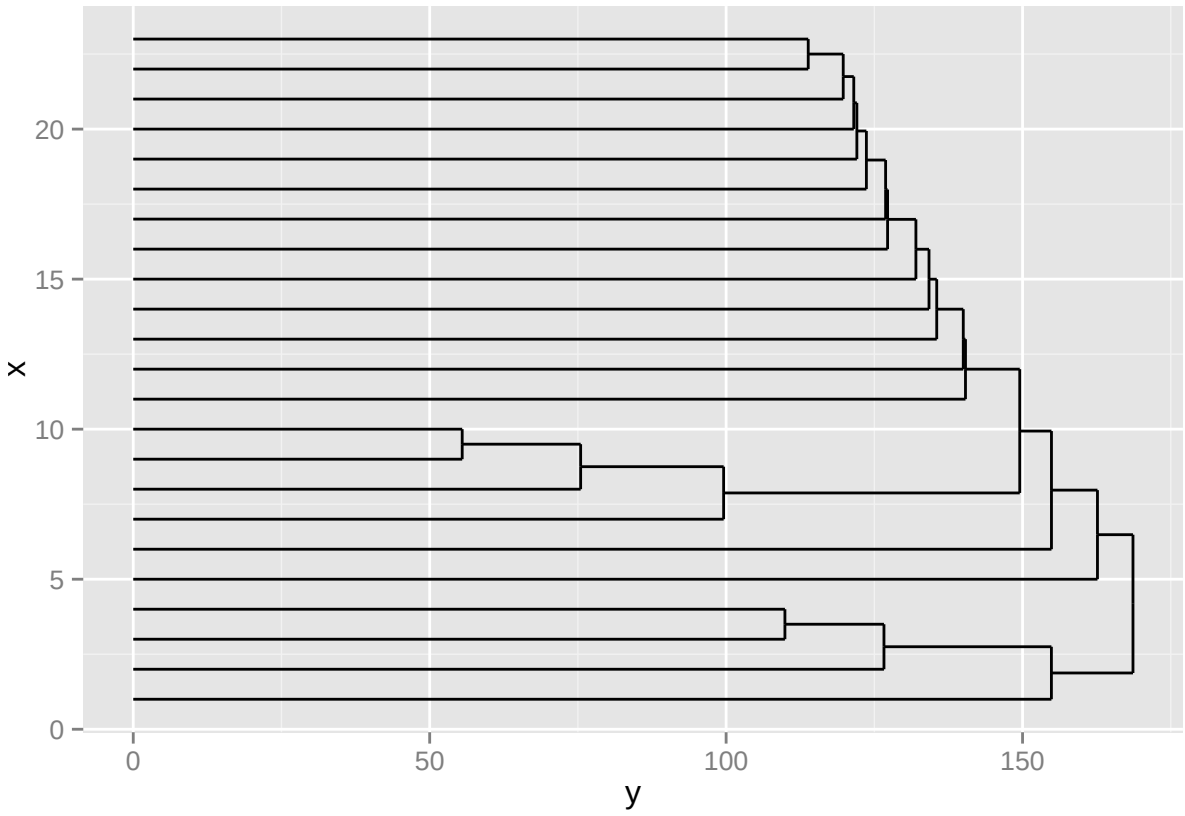
```
generateSubmission(LR_3_Pred, submitPath)
# testing  result: 0.90176
```

MODEL-4 (random forest on Bag of Words)

```
RF_4 = randomForest(Popular ~ ., data=newsTrainBoW[,!colnames(newsTrainBoW) %in% removedColumns], nodesi

trainPartition = createDataPartition(y=newsTrainBoW$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrainBoW[trainPartition,!colnames(newsTrainBoW) %in% removedColumns]
RF_4.tuned  = train(Popular ~ ., data=tuneTrain, method="rf", trControl=trainControl(method="cv", numbe

calcAUC(RF_4, newsTrainBoW$Popular)
# training result: 0.9399500

RF_4_Pred = predict(RF_4, newdata=newsTestBoW, type="prob")[,2]
generateSubmission(RF_4_Pred, submitPath)
# testing  result: 0.90048
```

Supervised + Unsupervised

```
matrixSparseText  = as.matrix(sparseText)
matrixSparseText.distMatrix = dist(scale(matrixSparseText))
matrixSparseText.clusters   = hclust(matrixSparseText.distMatrix, method="ward.D2")

dText = as.dendrogram(matrixSparseText.clusters)
dTextData <- dendro_data(dText, type = "rectangle")

ggplot(segment(dTextData)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend)) + coord_flip()
```




```
kText   = 25
mText   = t(sparseText)
KMCText = kmeans(mText, kText)

for (i in 1:kText) {
  cat(paste("cluster", i, ": ", sep=" "))
  s = sort(KMCText$centers[i, ], decreasing=TRUE)
  cat(names(s)[1:15], sep=", ", "\n")
}

newsData$TextCluster     = as.factor(KMCText$cluster)
newsData$PubDate = NULL
newsTrain = head(newsData, nrow(trainData))
newsTest  = tail(newsData, nrow(testData))
```

MODEL-5 (random forest on text cluster)

```
selectedColumns = cbind(selectedColumns, c("TextCluster"))
RF_5 = randomForest(Popular ~ ., data=newsTrain[, colnames(newsTrain) %in% selectedColumns], nodesize=5

trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, colnames(newsTrain) %in% selectedColumns]
RF_5.tuned  = train(Popular ~ ., data=tuneTrain, method="rf", trControl=trainControl(method="cv", numbe

calcAUC(RF_5, newsTrain$Popular)
# training result: 0.9399861
```

```
RF_5_Pred = predict(RF_5, newdata=newsTest, type="prob")[,2]
generateSubmission(RF_5_Pred, submitPath)
# testing  result: 0.89854
```

Features of negative example

```
newsData$QAboutNews = as.factor(ifelse(grepl("6q", newsData$Text) == TRUE, 1, 0))
newsData$DailyClip = as.factor(ifelse(grepl("DailyClipReport", newsData$Text) == TRUE, 1, 0))
newsData$DailyReport = as.factor(ifelse(grepl("DailyReport", newsData$Text) == TRUE, 1, 0))
newsData$PicOfDay = as.factor(ifelse(grepl("PicOfDay", newsData$Text) == TRUE, 1, 0))
newsData$TestYourself = as.factor(ifelse(grepl("TestYourself", newsData$Text) == TRUE, 1, 0))
newsData$WordofDay = as.factor(ifelse(grepl("WordofDay", newsData$Text) == TRUE, 1, 0))
newsData$Recap = as.factor(ifelse(grepl("recap", newsData$Text) == TRUE, 1, 0))

newsTrain = head(newsData, nrow(trainData))
newsTest  = tail(newsData, nrow(testData))
```

MODEL-6 (logistic regression with negative example features)

```
selectedColumns = cbind(selectedColumns, c("QAboutNews", "DailyClip", "DailyReport", "PicOfDay", "TestY

LR_6 = glm(Popular ~ ., data=newsTrain[, colnames(newsTrain) %in% selectedColumns], family=binomial)
calcAUClr(LR_6, newsTrain$Popular)

LR_6_Pred = predict(LR_6, newdata=newsTest[, colnames(newsTest) %in% selectedColumns], type="response")
generateSubmission(LR_6_Pred, submitPath)
```

MODEL-7 (random forest with negative example features)

```
RF_7 = randomForest(Popular ~ ., data=newsTrain[, colnames(newsTrain) %in% selectedColumns], nodesize=5

trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, colnames(newsTrain) %in% selectedColumns]

RF_7.tuned  = train(Popular ~ ., data=tuneTrain, method="rf", trControl=trainControl(method="cv", numbe

calcAUC(RF_7, newsTrain$Popular)
# training result: 0.9390981

RF_7_Pred = predict(RF_7, newdata=newsTest, type="prob")[,2]
generateSubmission(RF_7_Pred, submitPath)
# testing  result: 0.89867
```

Topic Words

Add some topic features according Topic from Google.

```
questionWords <- c("\\?","^why","should","^when","can","^if","^is")
religionWords <- c("secular","humanist","humanism","secularist","god","religion","atheist","atheism","i
               "islamists","church","atheists","jesus","christ","christian","catholic","pope","imam", "\
techWords <- c("apple","\\<ios\\>","ipod", "ipad","iphone")
```

```
healthWords <- c("cancer","weight","fat","heart","disease","brain","sex","sexual","love","hate","doctor
sciWords <- c("climate","warming","global","science","scientists")
busWords <- c("jobs","employment","work","working","economy")
poliHiWords <- c("republican", "conservative")

newsData$Question <- as.factor(ifelse(grepl(paste(questionWords, collapse="|"), newsData$Headline)==TRU
newsData$religionWords <- ifelse(grepl(paste(religionWords, collapse="|"), newsData$Headline)==TRUE,1,0
newsData$tech <- ifelse(grepl(paste(techWords, collapse="|"), newsData$Headline)==TRUE,1,0)
newsData$health <- ifelse(grepl(paste(healthWords, collapse="|"), newsData$Headline)==TRUE,1,0)
newsData$sci <- ifelse(grepl(paste(sciWords, collapse="|"), newsData$Headline)==TRUE,1,0)
newsData$business <- ifelse(grepl(paste(busWords, collapse="|"), newsData$Headline)==TRUE,1,0)
newsData$poliHi <- ifelse(grepl(paste(poliHiWords, collapse="|"), newsData$Headline)==TRUE,1,0)
newsData$noComment <- ifelse(grepl("no comment necessary", newsData$Headline),1,0)
newsData$comments <- ifelse(grepl("open for comments", newsData$Headline),1,0)
```

MODEL-8 (logistic regression with topic words)

```
LR_8 = glm(Popular ~ . - UniqueID, data=newsTrain, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
calcAUClr(LR_8, newsTrain$Popular)
# training result: 0.9412723

LR_8_Pred = predict(LR_8, newdata=newsTest, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
generateSubmission(LR_8_Pred, submitPath)
# testing  result: 0.89517
```

MODEL-9 (random forest with topic words)

```
RF_9 = randomForest(Popular ~ . - UniqueID, data=newsTrain, nodesize=5, ntree=2500, importance=TRUE)
trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, ]
RF_9.tuned  = train(Popular ~ . - UniqueID, data=tuneTrain, method="rf", trControl=trainControl(method=

calcAUC(RF_9, newsTrain$Popular)
# training result: 0.9418562

RF_9_Pred = predict(RF_9, newdata=newsTest, type="prob")[,2]
generateSubmission(RF_9_Pred, submitPath)
# testing  result: 0.89962
```

Model Evaluation

Model Name	Train ROC	Test ROC	DESC
LR_0	0.9315814	0.89088	logistic regression

Model Name	Train ROC	Test ROC	DESC
LR_1	0.9297705	0.89212	logistic regression on selected variables
LR_3	0.9389116	**0.90176**	logistic regression with bag of words
LR_6	0.9367199	0.90108	logistic regression with negative features
LR_8	0.9411255	0.89801	logistic regression with topic words
RF_2	0.9388249	0.89368	random forest
RF_4	0.9399030	0.89965	random forest with bag of words
RF_5	0.9387642	0.89865	random forest on text cluster
RF_7	0.9389404	0.89732	random forest with negative features
RF_9	0.941084	0.89887	random forest with topic words

Ensemble

```
EN_1 = (2*RF_9_Pred + 3*RF_4_Pred + 1*LR_3_Pred)/6
EN_2 = (2*RF_4_Pred + 3*LR_3_Pred + 1*LR_3_Pred)/6
EN_3 = (2*LR_1_Pred + 3*RF_2_Pred + 1*LR_8_Pred)/6
EN_4 = (2*LR_6_Pred + 3*RF_5_Pred + 1*RF_7_Pred)/6
EN_5 = (2*RF_5_Pred + 3*RF_4_Pred + 1*LR_3_Pred)/6
EN_6 = (1*RF_4_Pred + 2*LR_3_Pred + 1*LR_3_Pred)/4

generateSubmission(EN_1, submitPath)
generateSubmission(EN_2, submitPath)
generateSubmission(EN_3, submitPath)
generateSubmission(EN_4, submitPath)
generateSubmission(EN_5, submitPath)
generateSubmission(EN_6, submitPath)
```

Model Name	Test ROC
EN_1	0.90238
EN_2	**0.90550**
EN_3	0.89973
EN_4	0.90361
EN_5	0.90271
EN_6	0.90529